HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Etherlite
Date: June 1st, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Etherlite - Third Review
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Platform bridge with staking and rewards
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git commit	https://github.com/etherlite-org/pos-contracts/tree/25a1ed239d4fc1bee2069c1c811f81ec70ef8296/contracts
Timeline	20 MAY 2021 - 28 MAY 2021
Changelog	27 MAY 2021 - INITIAL AUDIT 28 MAY 2021 - SECOND REVIEW 1 JUNE 2021 - THIRD REVIEW

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Etherlite (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on June 1st, 2021.

Scope

The scope of the project is the smart contracts provided in the Git commit:

https://github.com/etherlite-org/pos-contracts/tree/25a1ed239d4fc1bee2069c1c811f81ec70ef8296/contracts

We have scanned these smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Asset’s integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured

Insecure	Poor secured	Secured	Well-secured

You are here

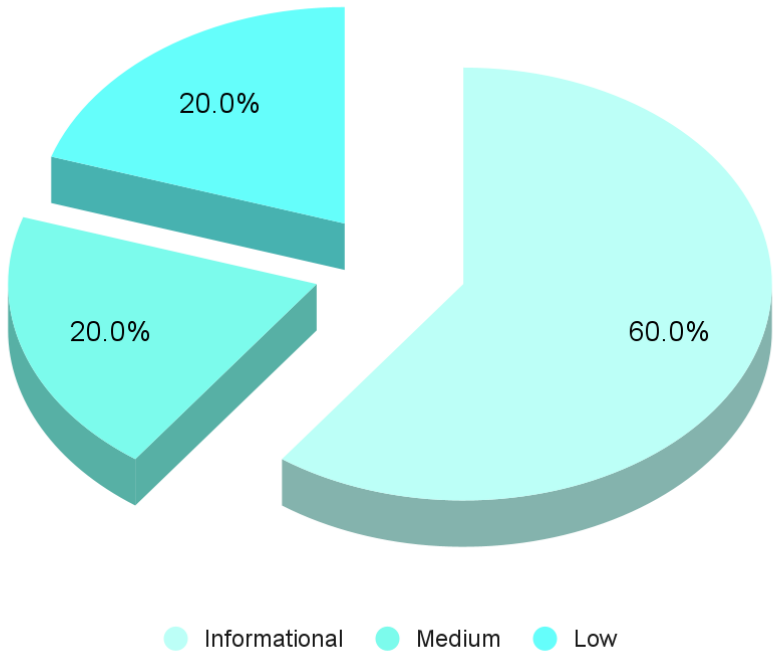
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

Security engineers found **1** medium, **1** low and **3** informational issues during the first review.

Security engineers found **1** low and **3** informational issues during the second review.

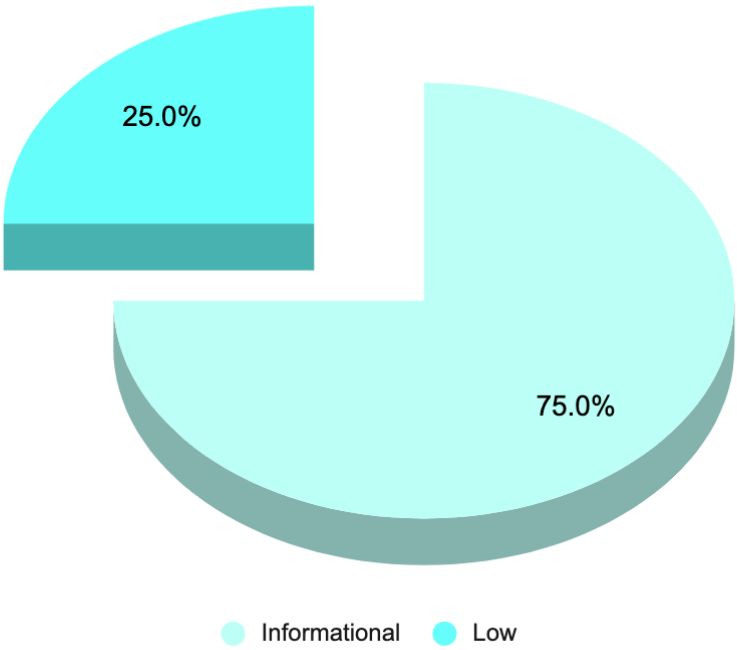
Security engineers found **2** informational issues during the third review.

Graph 1. The distribution of vulnerabilities after the first review.

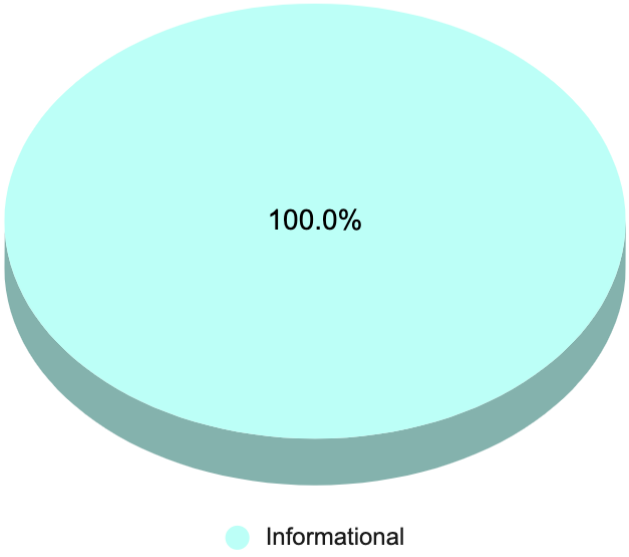


Graph 2. The distribution of vulnerabilities after the second review.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io



Graph 3. The distribution of vulnerabilities after the third review.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No Critical severity issues were found.

High

No High severity issues were found.

Medium

1. **Vulnerability:** Unused return

   The return value of the call to **mint** function is not used in the function logic.

   Fixed before second review

Low

1. **Vulnerability**: Unused function parameter

   Function parameter **uint256** _value is not being used, also the function does not override any other virtual function.

   Fixed before third review

Lowest / Code style / Best Practice

1. **Vulnerability:** Too many digits

   Literals with many digits are difficult to read and review.

   Recommendation: Please consider using ether units and/or scientific notation and/or separate with dashes

   ex:
   - 1_000_000
   - 1e6
   - 3.75 finney
   - 3750 szabo

   Lines: base/BlockRewardAuRaBase.sol#564

```
uint256 internal constant REWARD_PERCENT_MULTIPLIER = 1000000;
```

Lines: base/BlockRewardAuRaCoins.sol#16

```
uint256 public constant NATIVE_COIN_INFLATION_RATE = 3750000000000000;
```

Lines: TxPermission.sol#39

```
uint256 public constant BLOCK_GAS_LIMIT = 12500000;
```

Lines: TxPermission.sol#43

```
uint256 public constant BLOCK_GAS_LIMIT_REDUCED = 4000000;
```

2. **Vulnerability:** Unused state variable

   Internal constants CREATE and PRIVATE aren't used anywhere through the code

   Fixed before third review

3. **Vulnerability:** Public function that could be declared external

   public functions that are never called by the contract should be declared **external** to save gas.

   Lines: base/BlockRewardAuRaBase.sol#354

   ```
   function epochsPoolGotRewardFor(address _miningAddress) public view
   returns(uint256[] memory) {
   ```

   Lines: base/BlockRewardAuRaBase.sol#375

   ```
   function onTokenTransfer(address, uint256, bytes memory) public pure
   returns(bool) {
   ```

   Lines: base/BlockRewardAuRaBase.sol#383-386

   ```
   function epochsToClaimRewardFrom(
      address _poolStakingAddress,
      address _staker
   ) public view returns(uint256[] memory epochsToClaimFrom) {
   ```

   Lines: base/BlockRewardAuRaBase.sol#439

   ```
   function validatorRewardPercent(address _stakingAddress) public view
   returns(uint256) {
   ```

   Lines: RandomAuRa.sol#225

```
function getCipher(uint256 _collectRound, address _miningAddress)
public view returns(bytes memory) {
```

Lines: RandomAuRa.sol#241-244

```
function getCommitAndCipher(
   uint256 _collectRound,
   address _miningAddress
) public view returns(bytes32, bytes memory) {
```

Lines: RandomAuRa.sol#299

```
function nextCommitPhaseStartBlock() public view returns(uint256) {
```

Lines: RandomAuRa.sol#304

```
function nextRevealPhaseStartBlock() public view returns(uint256) {
```

Lines: RandomAuRa.sol#326

```
function revealSecretCallable(address _miningAddress, uint256 _number)
public view returns(bool) {
```

Lines: base/StakingAuRaBase.sol#381

```
function initialValidatorStake(uint256 _totalAmount) public onlyOwner {
```

Lines: base/StakingAuRaBase.sol#796

```
function poolDelegators(address _poolStakingAddress) public view
returns(address[] memory) {
```

Lines: base/StakingAuRaBase.sol#804

```
function poolDelegatorsInactive(address _poolStakingAddress) public
view returns(address[] memory) {
```

Lines: base/StakingAuRaBase.sol#822

```
function stakingEpochEndBlock() public view returns(uint256) {
```

Lines: base/StakingAuRaCoins.sol#184

```
function transferStakingAmount(uint256 _totalAmount) public payable{
```

Lines: TxPermission.sol#81

```
function addAllowedSender(address _sender) public onlyOwner
onlyInitialized {
```

Lines: TxPermission.sol#89

```
function removeAllowedSender(address _sender) public onlyOwner
onlyInitialized {
```

Lines: TxPermission.sol#113

```
function contractNameHash() public pure returns(bytes32) {
```

Lines: TxPermission.sol#118

```
function contractVersion() public pure returns(uint256) {
```

Lines: TxPermission.sol#125

```
function allowedSenders() public view returns(address[] memory) {
```

Lines: TxPermission.sol#145-155

```
function allowedTxTypes(
   address _sender,
   address _to,
   uint256 _value,
   uint256 _gasPrice,
   bytes memory _data
)
   public
   view
   returns(uint32 typesMask, bool cache)
{
```

Lines: TxPermission.sol#232

```
function blockGasLimit() public view returns(uint256) {
```

Lines: TxPriority.sol#55

```
function transferOwnership(address _newOwner) public onlyOwner {
```

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **1** medium, **1** low and **3** informational issues during the first review.

Security engineers found **1** low and **3** informational issues during the second review.

Security engineers found **2** informational issues during the third review.

Category	Check Items	Comments
➔ Code Review	➔ Gas Savings	➔ Public function that could be declared external
	➔ Style guide violation	➔ Too many digits

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.